CRAY RESEARCH, INC.

CRAY-1® COMPUTER SYSTEMS

UPDATE
REFERENCE MANUAL

SR-0013

Each time this manual is revised and reprinted, all changes issued against the previous version in the form of change packets are incorporated into the new version and the new version is assigned an alphabetic level. Between reprints, changes may be issued against the current version in the form of change packets. Each change packet is assigned a numeric designator, starting with 01 for the first change packet of each revision level.

Every page changed by a reprint or by a change packet has the revision level and change packet number in the lower righthand corner. Changes to part of a page are noted by a change bar along the margin of the page. A change bar in the margin opposite the page number indicates that the entire page is new; a dot in the same place indicates that information has been moved from one page to another, but has not otherwise changed.

Requests for copies of Cray Research, Inc. publications and comments about these publications should be directed to:

CRAY RESEARCH, INC.,
1440 Northland Drive,
Mendota Heights, Minnesota 55120

Revision	Description
	May, 1977 - Original printing.
A	June, 1979 - This printing represents a complete rewrite of the manual and brings it into agreement with release 1.06. Changes are not noted by change bars.
A-01	December, 1979 - This change packet brings the manual into agreement with release version 1.07. Changes are noted by change bars.
B	December, 1979 - This reprint includes change packet A-01. It contains no other changes.
B-01	April, 1980 - This change packet brings the manual into agreement with release version 1.08. Changes are noted by change bars.
C	November, 1980 - This reprint incorporates change packet B-01. It contains no technical changes. With this printing, the publication number has been changed from 2240013 to SR-0013.
D	June, 1981 - Rewrite. This printing is a complete rewrite of the manual, bringing the documentation into agreement with the 1.10 version of the released software. Major changes are the addition of the SQ control statement option, the NOSEQ directive, and the SEQ directive. Changes are not noted by change bars. This printing obsoletes all previous editions.

PREFACE

This manual describes UPDATE, a Cray Research, Inc. computer program that provides programmers with tools for modifying, editing, and updating source language programs on the CRAY-1 Operating System (COS). Use of this program provides the managing and tracking of software changes. UPDATE allows repeated results and simplifies the integration of separately produced changes into a single program. UPDATE executes under control of the CRAY-1 Operating System (COS) as described in the CRAY-OS Version 1 Reference Manual, publication SR-0011. The reader is assumed to be familiar with features of the operating system.

CONTENTS

INTRODUCTION 1

UPDATE is a system program that provides the user with a method of maintaining programs and other data on permanent datasets rather than on punched cards. These permanent datasets are called program libraries. A program library is created by UPDATE. The user can modify the program library by adding or purging decks or by adding or deleting (deactivating) cards from existing decks.

UPDATE can also be used to generate compile and source datasets. UPDATE executes on CRAY-1 Computer Systems under control of the CRAY-1 Operating System (COS).

The user calls UPDATE with an UPDATE control statement (see section 2).

PROGRAM LIBRARIES

A program library (PL) consists of one or more specially formatted card image decks, each separated by an end-of-file record. These decks can be programs, portions of programs, or input data for programs. UPDATE supports two types of decks: regular and common. A *regular* deck is a deck that is sequentially placed in the PL. It remains in only one location. A *common* deck is a deck that is sequentially placed in the PL but can be called in other locations of the PL. A common deck call can appear anywhere in a regular or common deck. (See the CALL directive, section 3.) During compile dataset generation, a common deck call is replaced by the text of the common deck.

Each deck, regular or common, has a unique identifier. At the end of all the decks in the PL, UPDATE supplies tables with descriptive information about each of the decks and the entire PL (figure 1-1). The information in these tables differs depending on the organization of the PL (random or sequential). (See Appendix C for information on PL formats.)

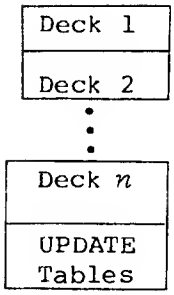


Figure 1-1. Sequence of decks and UPDATE tables in a program library

At the beginning of each card in a deck, UPDATE supplies descriptive information about the card. This descriptive information includes a card identifier that is a sequence number relative to the deck identifier. Figure 1-2 shows the card image of a program library.

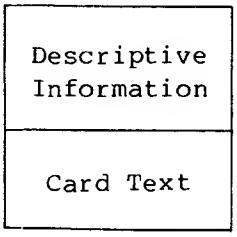


Figure 1-2. Program library card image

CREATION OF PROGRAM LIBRARIES

A program library is created from a collection of source decks either prepared by the user or previously generated by UPDATE as source output. The user creates a program library by supplying (1) an UPDATE statement, (2) UPDATE directives, and (3) input.

Input for PL creation can consist of one or more card decks. A creation run is signalled by P=0 on the UPDATE control statement. (Figure 1-3 shows a typical source deck input sequence.)

The UPDATE statement directs the computer to create a PL according to the specifications listed in the statement. (See section 2, The UPDATE Statement.)

The UPDATE directives specify whether the new deck will be a regular deck or common deck. This type of directive precedes each input deck. The programmer identifies and separates regular decks with DECK (DK) directives and common decks with COMDECK (CDK) directives. Other directives can be within the input decks to affect the output listing, call common decks, or write end-of-file records. (See section 3, UPDATE Directives.)

Note that because decks are preceded by either DECK or COMDECK directives and can have other UPDATE directives embedded in them, they should not be used as input to language processors such as the CAL assembler or the CFT compiler. Decks are not necessarily the equivalent of a program module in some source language. An UPDATE deck can be composed of only a small portion of a program, a complete program, data, or many programs, subroutines, or functions. Such delineations of code are entirely under the control of the UPDATE user.

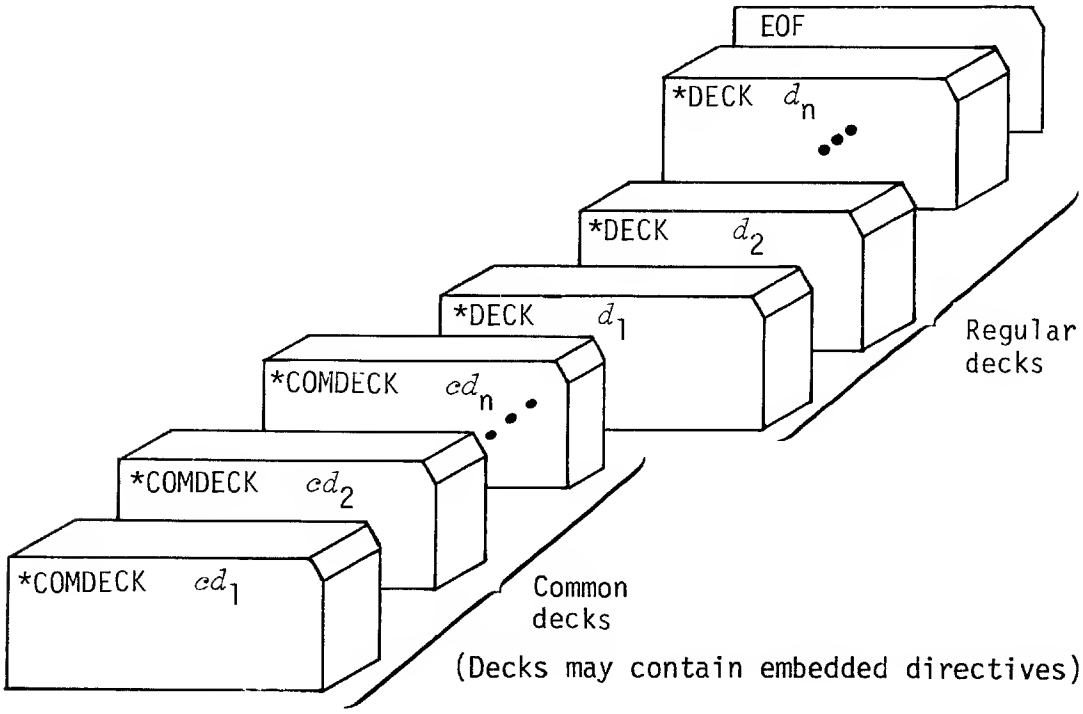


Figure 1-3. Typical source deck input sequence

PL MODIFICATION

Program libraries are modified by adding or purging decks, or by adding or deleting (deactivating) cards from existing decks. Program libraries are modified to produce a new program library, or an up-to-date compile dataset and/or source dataset. A modification set, comprised of directives for modifications and new text cards to be inserted, is assigned an identifier by the user.

PROGRAM LIBRARY FORMAT

In creating the program library, UPDATE records information according to decks, assigning each card in a deck a sequence number for later reference. Each card has a deck or common deck name and sequence number associated with it. The DECK or COMDECK directive has a sequence number of 1. Remaining cards in the deck are sequenced beginning with number 2. The next DECK or COMDECK terminates sequencing of one deck and begins sequencing of the next deck. An end-of-file or end-of-input terminates the final deck and input to UPDATE. The new program library consists of the sequenced decks in program library format followed by an identifier table. (See Appendix C, PL Format 2.)

PROGRAM LIBRARY RESTRICTIONS

The number of cards within one modification set or one deck can not exceed 131,071. The number of identifiers (i.e., modification set identifiers or deck identifiers) in one PL must not exceed 16,383.

UPDATE JOB FLOW

Figure 1-4 shows the execution of UPDATE. To create a PL, the user supplies UPDATE directives and source input in the form of card decks or a source deck.

Output from an UPDATE creation run can be selected listings, a compile or source dataset, or a new program library. The new program library is required output.

Input to an UPDATE modification run must include a program library and UPDATE directives and corrections. Source decks can also be used as input to modification runs.

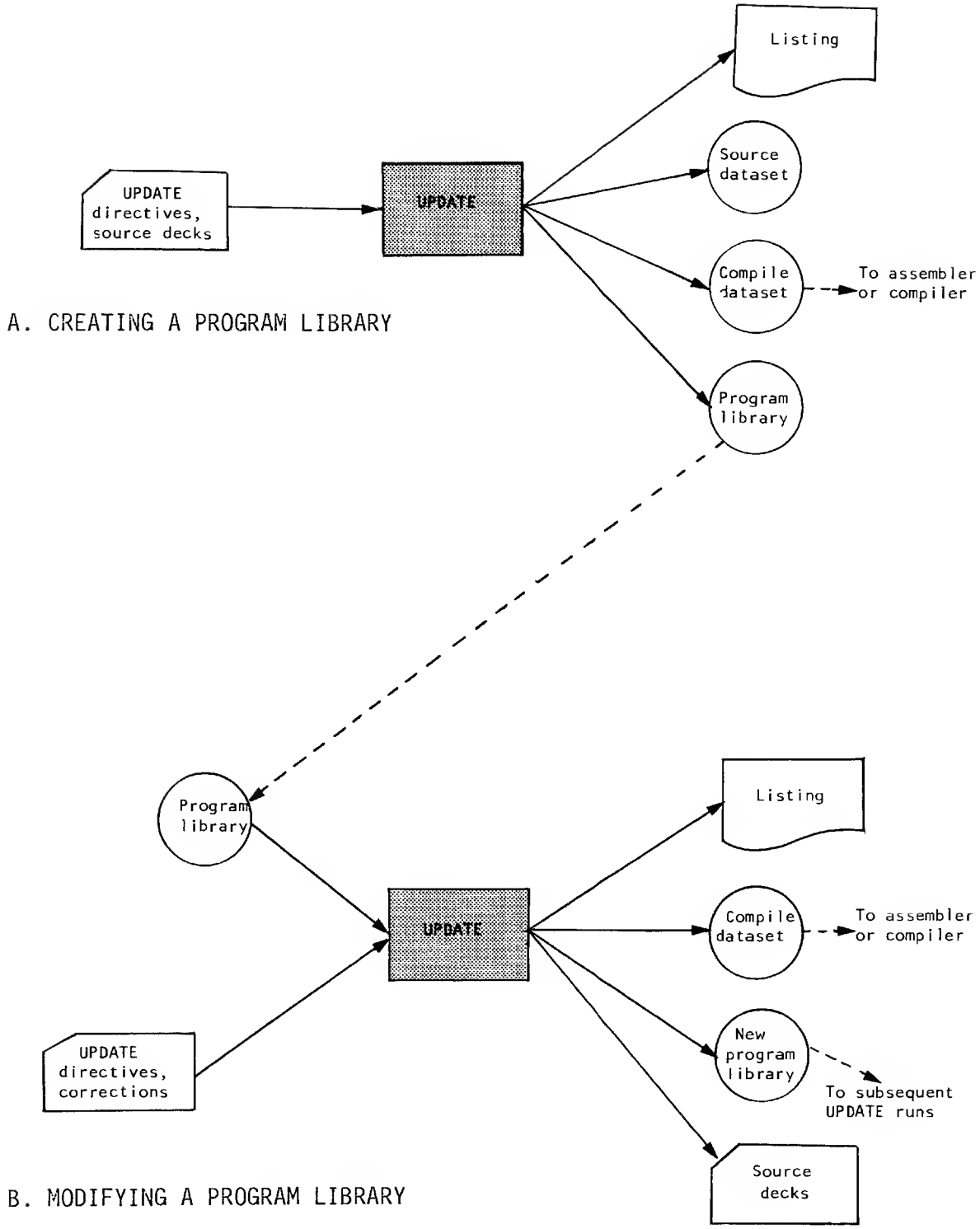


Figure 1-4. Data flow through UPDATE

Output from an UPDATE modification run can be a selected listing, a compile dataset, source decks, or a new program library.

Creation of a program library and modification of a program library must occur in separate UPDATE runs.

CONVENTIONS

The conventions used in this publication to describe statement and directive syntax are described below:

UPPERCASE	Identifies the command verb or literal parameter
UNDERLINED UPPERCASE	Specifies the minimum number of characters required for the verb or parameter to be recognized
Italics	Defines generic terms which represent the words or symbols to be supplied by the user
[] Brackets	Encloses optional portions of a command format
{ } Braces	Encloses two or more literal parameters when only one of the parameters must be used
... Ellipsis	Indicates optional use of the preceding item one or more times in succession

UPDATE CONTROL STATEMENT 2

The UPDATE statement causes the UPDATE program to be loaded into the user field and to begin execution. The control statement file portion of a user job deck must contain an UPDATE statement in the format described below. Parameters on the UPDATE statement specify datasets to be used, contents of the UPDATE listing, and other features of the run.

Conventions used in the UPDATE statement are described in section 1.

```
UPDATE,P=pldn,I=idn,C=cpldn,N=npldn,L=ldn,E=edn,
    S=sdn,*=m,/=c,DW=dw[,{F / Q=dk1:dk2:...:dkn}][,options].
```

P=*pldn* Program library dataset name.
If omitted or P, input PL is $PL.
If P=*pldn*, input program library is *pldn*.
If P=0, creation run only.

I=*idn* Input dataset; this dataset contains the directives and text for the UPDATE run.
If omitted or I, input dataset is next file of $IN.
If I=*idn*, input dataset is dataset having the name *idn*.
If I=0, illegal for creation run.

C=*cpldn* Compile output; decks determined by type of UPDATE run (F, Q, or normal) are written on dataset specified by this parameter.
If C or omitted, compile output is written to $CPL.
If C=*cpldn*, compile output is written to dataset *cpldn*.
If C=0, no compile output is generated.

N=*npldn* New program library name. Contents of the new program library are determined by the UPDATE mode (see table 2-1).
If omitted and in a *modification run*, no new program library is written.
If omitted and in a *creation run*, new program library is written to $NPL.
If N, new program library is written to $NPL.
If N=*npldn*, new program library is *npldn*.
If N=0, illegal for creation run.

L=*ldn* Listing dataset name; this dataset receives the UPDATE list output.
If omitted or L, list output is written to $OUT.
If L=*ldn*, list output is written to dataset named *ldn*.
If L=0, no list output is generated.

E=*edn* Error listing dataset name; this dataset contains directive/modification errors only.
If omitted or E, output is written to $OUT.
If E=*edn*, output is written to *edn*.
If E=0, errors are written to listing dataset only.

NOTE

If E and L specify the same dataset, L is honored and E is set to 0.

S=*sdn* Source dataset name; contents of this dataset are determined by the UPDATE mode (see table 2-1). This dataset can also be the input for a creation run.
If omitted or S=0, no source dataset is generated.
If S=*sdn*, source output is written to *sdn*.
If S, source output is written to $SR.

*=*m* Master character.
If omitted and in a *creation run*, master character for directives is *.
If omitted and in a *modification run*, master character is read from the PL.
If *=*m* and in a *creation run*, see Appendix A for legal master characters.
If *=*m* and in a *modification run*, *m* must match value read from the PL. *m* need not be specified on the control statement, however. Keyword only is illegal.

/=*c* Comment character.
If omitted, comment character on comment directive is /.
If /=*c*, comment character for comment directive is *c*.
Keyword only is illegal.

DW=*dw* Data width value. Columns 1 through *dw* of each compile output card image contain significant characters. Columns *dw*+1 through *dw*+15 are overwritten by spaces and sequencing information for compile output and, if the SQ option is in effect, source output. Source output records are truncated to *dw* columns. Input record columns *dw*+1 through 80 are overwritten by spaces.
If omitted or DW, columns 1-72 contain data.
If DW=*dw* or DW=L*dw*, columns 1-*dw* contain data. The *dw* range is 1-80.

NOTE

When data width value is omitted, DW is specified alone, or data width value is specified as *dw*; *dw*+1 through *dw*+8 contain an identifier, right-justified with leading spaces; *dw*+9 contains a period; and *dw*+10 through *dw*+15 contain a sequence number, left-justified with trailing spaces.

When data width value is specified as L*dw*, the entire compile output sequencing field is left-justified.

F, Q, or omitted
Full, quick, or normal UPDATE run. Specifies contents of compile, and/or source datasets, and the new PL. (See table 2-1.)

F Full UPDATE mode. All active cards in the PL are written to compile and/or source datasets. Sequence is determined by the PL identifier table. No COMPILE directive is necessary.

Q=dk_1:dk_2:...:dk_n
: Quick UPDATE mode. Decks specified with the Q parameter and decks specified by a COMPILE directive are written to compile and/or source datasets and the new PL. Sequence is determined by the PL identifier table. Corrections to decks not specified with the Q parameter or by a COMPILE directive are not included. Up to 100 decks can be specified in this manner. Deck names that are unknown after all input has been entered are errors.

omitted
: Normal UPDATE mode. Decks specified by COMPILE directives, modified decks, and decks calling modified common decks are written to compile and/or source datasets.

options (Keyword only)
The following options are available on the control statement:

- NA Do not abort if directive/modification errors occur. All requested datasets are generated.
- NR Do not rewind source/compile datasets at beginning or end of UPDATE execution.
- IN List input to *ldn*.
- ID Write identifier summary to *ldn*.
- ED Write edited card summary to *ldn*.
- CD Write compile dataset generation directives to *ldn*.
- UM Write unprocessed modifications to listing and/or error datasets. Ignored in full or normal mode.
- K Order decks written to compile and new PL datasets as directed by Q control statement parameter values and COMPILE directives. Ignored in full or normal mode.
- SQ UPDATE source output provided with sequencing beginning in column *dw*+1. SQ has no effect on compile dataset output.

NOTE

If a modification set affects two or more decks and the K option is in effect, the sequence numbers of inserted cards can be inconsistent with sequencing that has occurred without the K option.

Table 2-1. Dataset contents for full, quick, and normal mode

Mode	Compile Dataset Contents (Embedded directives processed)	Source Dataset Contents (Embedded directives considered text but not processed)	New PL Dataset Contents (Embedded directives written to new PL)
Full	All decks and called common decks	All decks and all common decks	All decks and all common decks
Quick	Only decks specified by COMPILE directives and UPDATE control statement parameter Q and called common decks	Same as compile dataset contents	Only decks specified by COMPILE directives and UPDATE control statement parameter Q and all common decks
Normal	Decks specified by COMPILE directives, modified decks, and decks calling modified common decks	Same as COMPILE dataset contents	Same as full mode

Examples:

The following example shows a program library creation.

```
UPDATE,P=0,N=NEWPL,I=INPUT.
```

In this example:

- P=0 specifies no existing program library.
- New PL will be written to NEWPL.
- Input is read from INPUT.
- Compile output is written to $CPL; all decks are selected.

This example shows a program library modification:

```
UPDATE,P,I=MODS,Q=DECK3:DECK2:DECK4,K,NR,NA.
```

In this example:

- P=$PL is implied.
- Input is read from MODS.
- Quick mode with K option. Assuming a single COMPILE directive (*COMPILE DECK1.DECK4), DECK1 through DECK4 are written to $CPL in the following order:
  DECK3
  DECK2
  DECK4
  DECK1
- $CPL is not rewound by UPDATE before or after execution.
- UPDATE does not abort if directive or modification errors occur.

UPDATE DIRECTIVES 3

UPDATE recognizes a set of directives that direct it in its task of modifying decks and generating compile output and program libraries. Directives usually occupy a file of the input dataset ($IN) but can reside on a separate dataset.

This section gives the general format and rules for directives and describes the directives in alphabetical order. In this manual, * is used as the master character for directive descriptions. Conventions used in UPDATE directives are described in section 1.

SYNTAX

A directive has the following syntax:

md,p_1,p_2,p_3,... *<comment>*

Parameters:

m	PL master character
d	Directive name or abbreviated name
p_i	Parameters dependent on directive
<comment>	Optional comment

The first comma can be replaced with one or more spaces. The comment, if present, must be preceded by one or more spaces.

The underlined uppercase format of the directives specifies the minimum number of characters required for the verb or parameter to be recognized.

CARD IDENTIFICATION

Each modification set and each deck (or common deck) has a unique identifier associated with it. This identifier is the name from the corresponding DECK, COMDECK, or IDENT directive card.

The sequence number for a deck (or common deck) card is derived from the position of the card within the deck. The card sequence number of a given card inserted by a modification set is usually not immediately known. Sequencing depends on the ordering of existing cards within the PL.

A given card is referred to by *id.seq*, where

id is the deck or modification set identifier name, and
seq is the card sequence number.

Once a deck (or common deck) or modification set becomes a part of a new program library, *id.seq* is permanent.

IDENTIFIER NAMES

Each identifier (deck, common deck, or ident name) is a 1- to 8-character name assigned when the identifier is first used. Names cannot include commas, periods, blanks, colons, or equal signs but can include any other character in the ASCII code range of 41_8 through 176_8 (see Appendix A).

On some directives, names can be specified individually or as an inclusive range. Where a range of names is specified, the parameter consists of the name of the first identifier in the range, a period, and the final identifier in the range. The format is as follows:

deckname (first).*deckname* (last)

DIRECTIVE EXAMPLES

```
*INSERT MOD1.79

*D X.76,Y.79

$/                      $ is master character; / is comment character.

@EDIT DECK1.DECK2       @ is master character.

$$                      $ is master and comment character.

%BEFORE,X.23            % is master character.
```

DIRECTIVES

The set of directives recognized by UPDATE follows in alphabetical order:

BEFORE - INSERT BEFORE DIRECTIVE

The BEFORE directive indicates that cards immediately following it are to be inserted before the card specified.

Format:

```
*BEFORE id.seq
```

id Deck or modification set identifier name

seq Card sequence number

CALL - CALL COMMON DECK DIRECTIVE

The CALL directive indicates the location at which a common deck is to be placed when the compile dataset is generated. The combination of common decks and CALL directives allows a user to maintain a single copy of common text and be assured that the most up-to-date copy is always used in a deck that calls it. A common deck can contain CALL directives for other common decks but must not contain a CALL for itself. The CALL directive is embedded in a deck, common deck, or input text, and is assigned a sequence number accordingly.

Format:

```
*CALL comdeck
```

comdeck Name of the common deck

COMDECK - COMMON DECK DIRECTIVE

The COMDECK directive introduces a common deck. Cards up to the next DECK, COMDECK, IDENT, INSERT, DELETE, BEFORE, or end of input comprise the common deck. Other directives are interpreted but do not terminate the common deck.

The COMDECK directive is the first card of the common deck and is assigned sequence number 1.

Format:

```
*COMDECK,cmdk,NOPROP
```

cmdk Name of the common deck

NOPROP No propagation parameter (optional). If specified, this parameter indicates to UPDATE that decks calling this deck are not to be automatically considered as modified whenever this common deck is modified. If omitted, and the common deck is modified, all decks containing CALLS for this common deck are also considered modified.

COMPILE - COMPILE DIRECTIVE

COMPILE directives specify the contents of the compile and/or source dataset(s). When selecting decks for compile output, called common decks need not be specified. Generating source output requires all desired common decks to be specified.

COMPILE directives can occur anywhere in the input but cannot be used to refer to unknown decks.

Format:

```
*COMPILE p1,p2,...,pj.pk,...,pn
```

p_1 Single deck name or a common deck name

$p_j p_k$ Range of deck names and/or common deck names. Parameter order is significant only when the K control statement parameter is selected. Parameter order then specifies deck order for the compile and new PL datasets. The decks are otherwise written in the order they appear in the PL identifier table.

CWEOF - CONDITIONAL WRITE END-OF-FILE DIRECTIVE

The CWEOF directive directs UPDATE to write an end-of-file to the compile dataset if (1) the compile dataset was not positioned after an end-of-file and (2) the compile dataset is not at beginning of data.

The CWEOF directive is embedded in a deck, common deck, or input text, and is assigned a sequence number accordingly.

Format:

```
*CWEOF
```

The CWEOF directive is ignored if no compile output is requested.

DECK - DECK DIRECTIVE

The DECK directive introduces a new deck. Cards up to the next DECK, COMDECK, IDENT, INSERT, DELETE, BEFORE, or end of input comprise the deck. The new deck is placed in the new program library at the end of the existing decks. Decks can contain embedded directives (CALL, CWEOF or WEOF). Other directives are interpreted but do not terminate the deck.

The DECK directive is the first card of the deck and is assigned sequence number 1.

Format:

```
*DECK deck
```

deck Name of the new deck

DELETE - DELETE CARDS DIRECTIVE

The DELETE directive allows a user to delete (deactivate) cards or ranges of cards and optionally replace them with cards appearing after the DELETE directive.

A deleted card is copied to the new program library. It retains its identification but is flagged as inactive. Inactive cards are not included in compile and source output. A deletion range must not cross a deck boundary.

Formats:

*DELETE $id_1.seq_1,id_2.seq_2$	(range delete)
*DELETE $id_1.seq_1,seq_2$	(range delete, short form)
*DELETE $id_1.seq_1$	(single card delete)

id_i k or modification set identifier name

seq_1 and seq_2
Card sequence numbers

EDIT - EDIT DECKS DIRECTIVE

The EDIT directive removes all inactive cards from decks and/or common decks. No resequencing is performed. UPDATE edits only those decks noted explicitly on the EDIT directive.

Format:

*EDIT $p_1,p_2,\ldots,p_j.p_k,\ldots,p_n$

p_1 Single deck or common deck

$p_j.p_k$ Range of decks and/or common decks

IDENT - MODIFICATION SET IDENTIFICATION DIRECTIVE

The first card in a modification set is an IDENT directive. This directive provides the modification set identifier that is to be associated with all of the changes. When no new program library is being generated, this directive is optional. The default identifier is *.NOID.*.

Format:

*IDENT *ident*,U=id_1:id_2:id_3:...:id_n,K=id_4:id_5:id_6:...:id_m

ident Modification set identifier

U=*id* Unknown modification ids; specifies an unknown ID dependency. This modification set is ignored if UPDATE finds the specified ID in its list of IDs on the program library. An identifier is unknown if it is not the name of a deck or modification set identifier.

K=*id* Known modification ids; specifies a dependency ID. This modification set is ignored if UPDATE cannnot find the specified ID in its list of IDs on the program library.

If all dependencies are not met, UPDATE skips the modification set, the modification identifier remains unknown, and an informative logfile message is issued.

U and K arguments can also be shown as

U=id_1,U=id_2,...K=id_6,...

Example:

The following directive assigns the ID MOD84 to the modification set. UPDATE processes the modification set only if MOD83 is known and MOD85 is unknown.

```
*IDENT MOD84,K=MOD83,U=MOD85
```

INSERT - INSERT AFTER DIRECTIVE

The INSERT directive indicates that cards immediately following it are to be inserted after the card specified.

Format:

```
*INSERT id.seq
```

id Deck or modification set identifier name

seq Card sequence number

LIST AND NOLIST - RESUME/STOP LISTING DIRECTIVES

LIST and NOLIST resume the listing or stop the listing, respectively, of cards in the input stream. These directives can occur anywhere in the input. They control the input listing but are otherwise ignored.

The L=0 UPDATE statement parameter overrides the LIST directive. The NOLIST directive overrides the UPDATE statement option IN.

Formats:

```
*LIST

*NOLIST
```

MOVEDK - MOVE DECK DIRECTIVE

The MOVEDK directive causes UPDATE to move an entire deck from its present location to a point immediately following a specified destination deck. The sequencing information within the moved deck is not altered.

Formats:

```
*MOVEDK dk1:dk2
*MOVEDK dk1:.
```

dk_1 Deck or common deck to be moved

dk_2 Destination deck or common deck

. Specifies beginning of the PL

The moved deck resides at the indicated point immediately after this directive is successfully processed by UPDATE.

PURGEDK - REMOVE DECK DIRECTIVE

The PURGEDK directive instructs UPDATE to immediately remove the entire deck from the PL.

Format:

```
*PURGEDK dk
```

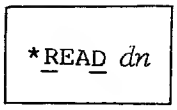
dk Name of the deck or common deck to be removed

READ - READ ALTERNATE INPUT DIRECTIVE

The READ directive causes UPDATE to begin reading input from the named dataset starting at its current position. An end-of-file on an input dataset causes UPDATE to resume reading from the previous dataset. READ directives can appear anywhere but must not be recursive.

If the dataset has not been accessed before UPDATE execution, UPDATE will attempt to access the permanent dataset with all defaults. (See the CRAY-OS Version 1 Reference Manual.)

Format:

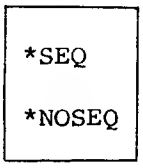


dn Name of the dataset

SEQ AND NOSEQ - START/STOP SEQUENCE NUMBER WRITING DIRECTIVES

SEQ and NOSEQ begin writing or stop writing, respectively, of sequence numbers to the compile dataset. The compile dataset output contains *dw* data columns per record. The SEQ and NOSEQ directives are ignored if no compile output is requested.

Formats:



WEOF - WRITE END-OF-FILE DIRECTIVE

A WEOF directive causes UPDATE to write an end-of-file to the compile dataset. The WEOF directive is embedded in a deck, common deck, or input text, and is assigned a sequence number accordingly.

Format:

The WEOF directive is ignored if no compile output is requested.

/ - COMMENT DIRECTIVE

A comment directive is copied to the list output.

Format:

```
*/ optional comment
```

The comment character is taken from the UPDATE statement.

MODIFICATION OF A PROGRAM LIBRARY 4

OVERVIEW

The user can modify the PL by adding or purging decks or by adding or deleting (deactivating) cards from existing decks. Modifications are applied against an existing program library to produce a new program library, or an up-to-date compile dataset, and/or source dataset. A modification set is comprised of directives for modifications and new text cards to be inserted. Each modification set is assigned an identifier by the user.

NEW PL GENERATION

To modify a PL, the user supplies (1) an UPDATE statement directing the computer to modify that PL; (2) UPDATE directives specifying the modification set identifier or the deck identifier, cards to be deleted and/or inserted, and the locations of the modifications; and (3) any new cards to be added. If the new cards to be added are on another dataset, they can be read from that dataset. When a PL has been modified, the newly generated dataset is known as a *new PL*.

Figure 4-1 illustrates input for a modification run. Input for a modification run can consist of a modification set and/or new decks to be added. A modification set is preceded by a modification set identifier; new decks are preceded by deck identifiers. If text or a reference to text is to be inserted at a location specified in the directive, it must immediately follow the directive. Decks can also contain certain directives such as CALL, CWEOF, and WEOF. See examples 1 and 2 following.

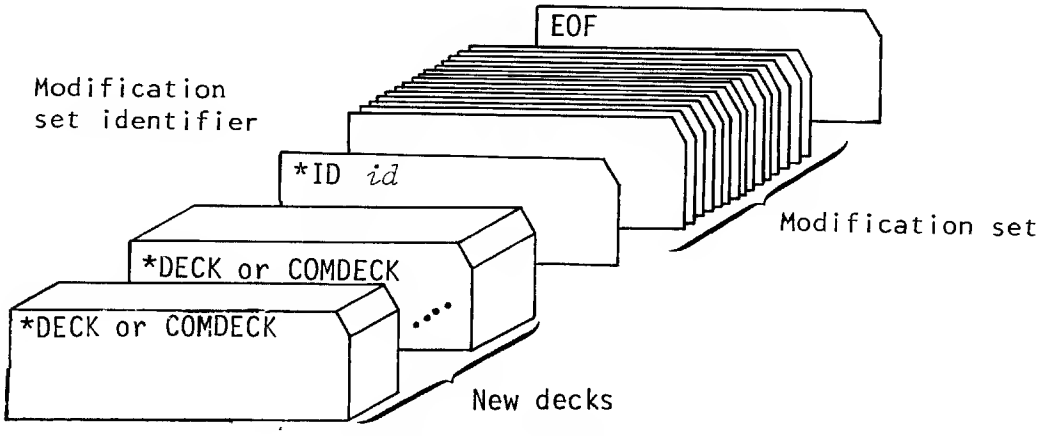


Figure 4-1. Sample PL modification set

Following a modification run, the new PL (if generated) consists of corrected decks and an updated identifier table. (See Appendix C, PL Format 2.)

Example 1:

This example shows a modification of program library LIB. The modification set identifier is PLMOD2. Each card to be inserted has the name PLMOD2 and a sequence number. (See section 3, UPDATE Directives, for a description of the directives.)

```
JOB,JN=UP1.
ACCESS,DN=$PL,PDN=LIB.
UPDATE,N.
SAVE,DN=$NPL,PDN=LIB2.
  .
  .
  .
eof
*ID PLMOD2
*D ABC.3
  (cards to be inserted here)
*B XYZ.5
  (cards to be inserted here)
*I ABC.7
  (cards to be inserted here)
eof
```

Example 2:

This example shows a modification of program library MYPL. Input is read from dataset LIBDEK. Card ABC.15 is a part of the PL. The cards inserted after ABC.15 and after the deletion of DEK.5 will have the name PLMOD3 and sequence numbers relative to their placement in the PL. The Q parameter in the UPDATE statement indicates that only deck DEK will be written to the compile dataset and the new PL.

```
JOB,JN=UP2.
ACCESS,DN=$PL,PDN=MYPL.
ACCESS,DN=LIBDEK.
UPDATE,N,Q=DEK.
SAVE,DN=$NPL,PDN=PLMOD3.
  .
  .
  .
eof
*READ LIBDEK
*ID PLMOD3
*INSERT  ABC.15
  (cards to be inserted here)
*D DEK.5
  (cards to be inserted here)
eof
```

COMPILE AND SOURCE DATASETS GENERATION

The user can also generate compile and source datasets from program libraries. A *compile* dataset contains selected PL decks that have been stripped of UPDATE directives, deactivated cards, and UPDATE tables (see example 3). A compile dataset can be compiled or assembled. Each compile dataset record is *dw*+15 characters long (see the DW parameter, section 2).

A *source* dataset contains selected PL decks in card image form. These decks have been stripped of card descriptive information, deactivated cards, and UPDATE tables. The decks retain any embedded directives (see example 4). A source dataset is a current, edited copy of a PL or a deck. This type of dataset cannot be compiled or assembled because of embedded directives. However, it can be used to add a new deck to a PL. Each source dataset record is *dw* characters long (see the DW parameter, section 2).

Examples 3 and 4 show the extraction of deck AB for compile and source datasets, respectively. No modifications were made.

Example 3:

```
JOB,JN=UP3.
ACCESS,DN=$PL,PDN=LIB3.
UPDATE.
SAVE,DN=$CPL,PDN=CMP.
  .
  .
  .
eof
*C AB
eof
```

Example 4:

```
JOB,JN=UP3.
ACCESS,DN=$PL,PDN=LIB3.
UPDATE,S,C=0.
SAVE,DN=$SR,PDN=SRC.
.
.
.
eof
*C AB
eof
```

MODIFICATION PROCESS

UPDATE processes the modifications in two passes. During the first pass, it scans the directives and new text and constructs tables for use during the second pass.

During the second pass, UPDATE modifies each deck on a deck-by-deck basis, deleting cards and sequencing and identifying each new card according to its modification set identifier.

Directives can cause card images to be inserted or deleted from the program library. A deletion does not physically remove the card image but deactivates it instead. In its deactivated state, it is logically removed and does not appear on compile or source output. UPDATE maintains a record of active and inactive cards. (See Appendix C, PL Format 2.) If the card status bit indicates that the card is inactive, the card has been deleted and is, in effect, removed from the deck. If the status of the card is active, the card is in the deck.

If compile output is desired from a modification run, UPDATE creates it during the second pass. Calls for common decks are replaced with copies of the common deck at this time. The calls may have been in the decks originally or inserted during a modification run. Other embedded directives are also executed at this time.

APPENDIX SECTION

CHARACTER SET

UPDATE Character	Code	UPDATE Character	Code	UPDATE Character	Code
Space	040 (sep)	@	100	_	137
!	041	A	101	'	140
"	042	B	102	a	141
#	043	C	103	b	142
$	044	D	104	c	143
%	045	E	105	d	144
&	046	F	106	e	145
'	047	G	107	f	146
(	050	H	110	g	147
)	051	I	111	h	150
*	052	J	112	i	151
+	053	K	113	j	152
,	054 (sep)	L	114	k	153
-	055	M	115	l	154
.	056 (sep)	N	116	m	155
/	057	O	117	n	156
0	060	P	120	o	157
1	061	Q	121	p	160
2	062	R	122	q	161
3	063	S	123	r	162
4	064	T	124	s	163
5	065	U	125	t	164
6	066	V	126	u	165
7	067	W	127	v	166
8	070	X	130	w	167

UPDATE Character	Code	UPDATE Character	Code	UPDATE Character	Code
9	071	Y	131	x	170
:	072 (sep)	Z	132	y	171
;	073	[	133	z	172
<	074	\	134	{	173
=	075 (sep)	]	135	¦	174
>	076	^	136	}	175
?	077			~	176

Characters NUL through US and DEL (code 000 through 037 and 177) are not recognized. Separators (sep) are illegal name, master, and comment characters.

UPDATE MESSAGES

B

The UPDATE program generates three types of logfile messages:

- Informative - No action is taken
- Error - Aborts job when UPDATE execution is finished unless the UPDATE statement parameter NA is selected
- Fatal Error - Aborts execution immediately

Messages are preceded with a code identifier as shown below.

Informative Messages	Error Messages	Fatal Error Messages
UD001	UD005	UD004
UD002	UD006	UD007
UD003	UD008	UD009
UD011	UD015	UD010
UD012	UD020	UD013
UD016		UD014
UD017		
UD018		
UD019		
UD021		

Messages listed in this section are in numeric sequence by code identifier. An explanation follows each message.

UD001 - PL DATE *mm/dd/yy* LEVEL: *name*

mm/dd/yy indicates the creation date of the PL and *name* is the name of the last identifier recorded.

UD002 - *n* WARNING ERRORS

UD003 - EMPTY INPUT FILE, DN = *dn*

UD004 - DATASET NOT LOCAL, DN = *dn*

The dataset indicated was not accessed prior to UPDATE execution.

UD005 - RECURSIVE READ, DN = *dn*

An attempt was made to read dataset *dn* recursively.

UD006 - ILLEGAL READ DSNAME: *name*

A READ directive encountered by UPDATE while reading input contains an illegal dataset name.

UD007 - CONTROL STATEMENT ERROR

One of the following control statement errors exists:

- New PL and PL datasets have the same name
- Both F and Q were specified
- P=0 and I=0 (initial PL generation, but no input)
- Illegal comment and/or master character
- Illegal DW value

UD008 - *ERROR* MODS W/OUT ID - N=0

A new PL cannot be generated with mods that are not identified with an IDENT directive. New PL generation has been suppressed.

UD009 - MASTER CHARACTER MISMATCH

The master character supplied with the control statement does not match the master character recorded in the PL.

UD010 - P IS NOT A PROGRAM LIBRARY

UD011 - *n* UPDATE ERRORS

UD012 - PL FORMAT CONVERSION COMPLETE

A sequential format PL has been internally rewritten as a random format PL. (See Appendix C.)

UD013 - SEQ. NO. > 131071 DIVIDE TEXT

An attempt was made to add more than 131,071 cards with one identifier. The insertion must be split over two or more identifiers.

UD014 - NUMBER OF IDENTS > 16383

Too many DK/CDK/ID names in PL.

UD015 - UNKNOWN "Q" DK: *name* *ERROR*

name listed as a parameter value for the Q keyword is not a deck or common deck.

UD016 - PL MASTER CHARACTER IS: *m*

m is the master character recorded when the PL was built.

UD017 - MOD ID SKIPPED: *name*

Modification set *name* was skipped due to unsatisfied IDENT directive dependency conditions.

UD018 - *n* UNPROCESSED MODS

References were made to nonexistent cards. UPDATE finished execution without processing *n* modification directions.

UD019 - *n* LONG INPUT RECORDS

n input records greater than 80 columns were truncated at column 80.

UD020 - EXCESSIVE INPUT ERRORS - ABORT

More than 100 input errors were detected. UPDATE aborted. A DECK or COMDECK directive may be missing.

UD021 - *n* OVERLAPPING MODS

Inserted cards were referenced by subsequent mods. Check input to determine if the overlaps were proper and expected.

UPDATE PROGRAM LIBRARY FORMATS C

UPDATE accepts two program library formats. Format 1 program libraries can be read only. These libraries were created with UPDATE version 1.05 or earlier but are still available to the user. As a preliminary step, UPDATE always internally converts PL format 1 to PL format 2. When new program libraries are written, the format 2 structure is always used. Formats are completely under control of UPDATE.

FORMAT 1 - SEQUENTIAL PL STRUCTURE

Format 1 (figure C-1) is a sequential PL structure. Each section of the PL (decks and lists) is separated by an *eof*. Decks consist of individual cards images that are not separated by *eor*. Details of each format follows.

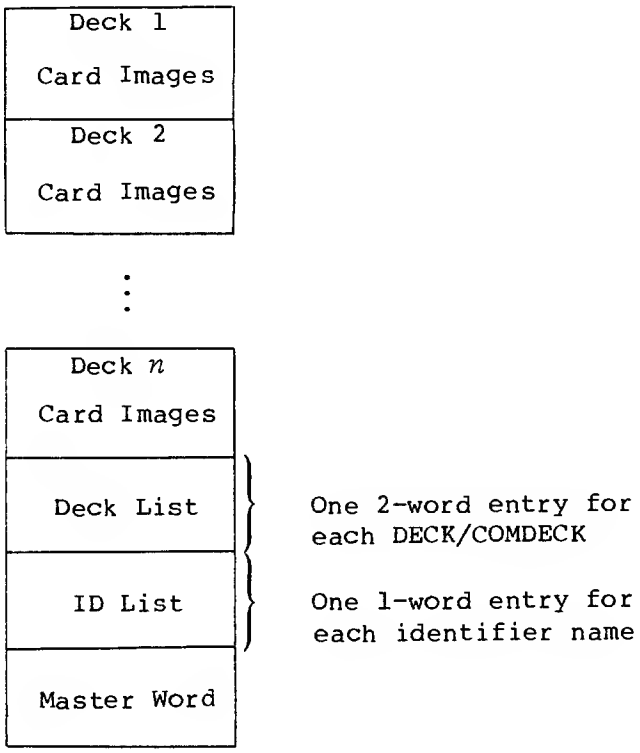
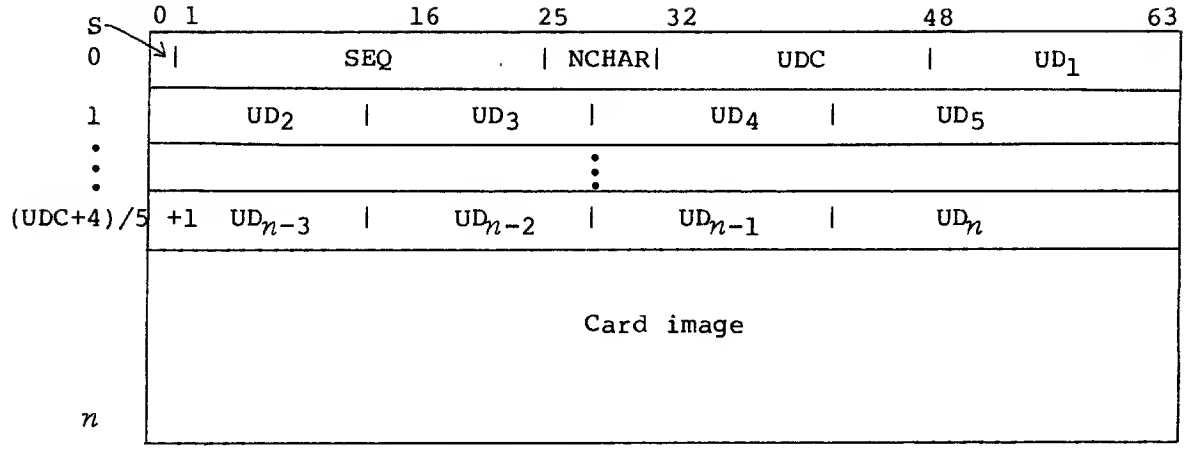


Figure C-1. PL format 1

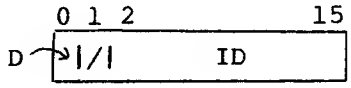
PL Card Format:



Field	Words	Bits	Description
S	0	0	Card status bit: 0 Inactive 1 Active
SEQ	0	1-24	Sequence number
NCHAR	0	25-31	Number of characters in card text
UDC	0	32-47	Descriptor count
UD_1	0	48-63	First UPDATE descriptor (identifies name that introduces the card)
UD_i-UD_n	1-(UDC+4)/5	+1	Modification descriptors identifying deletion identifier name

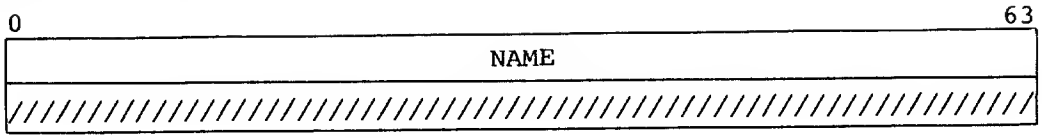
The length of the card image is NCHAR bytes.

Descriptor Format:



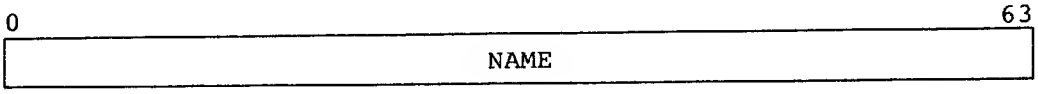
Field	Bits	Description
D	0	Descriptor status: 0 Deactivated card 1 Activated card
	1	Reserved
ID	2-15	Identifier number or name

Deck List Entry Format:



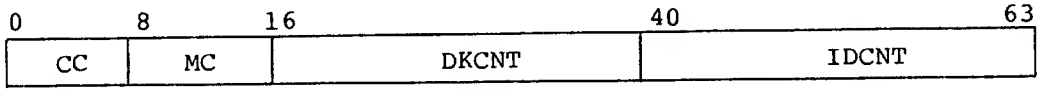
Field	Bits	Description
NAME	0-63	Name of DECK or COMDECK (left-justified with space fill)
(word 2)		Reserved

ID List Entry Format:



Field	Bits	Description
Name	0-63	Identifier name (left-justified with space fill)

Master Word Format:



Field	Bits	Description
CC	0-7	PL check character (O'141 = lowercase A)
MC	8-15	PL master character
DKCNT	16-39	Length of the deck list
IDCNT	40-63	Length of the ID list

FORMAT 2 - RANDOM PL STRUCTURE

Format 2 (figure C-2) is a random PL structure. Each section of the PL is separated by *eof* records. Card images within each deck are separated by *eor*. The identifier table and PL information contain no *eor*. Details of each format follows.

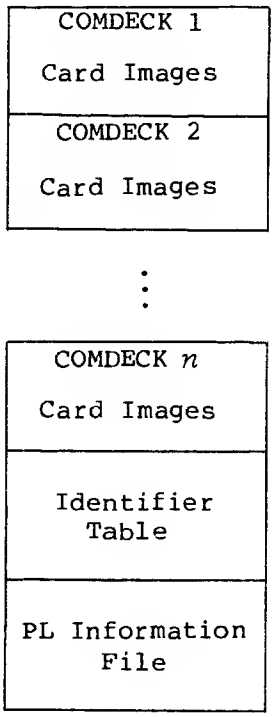
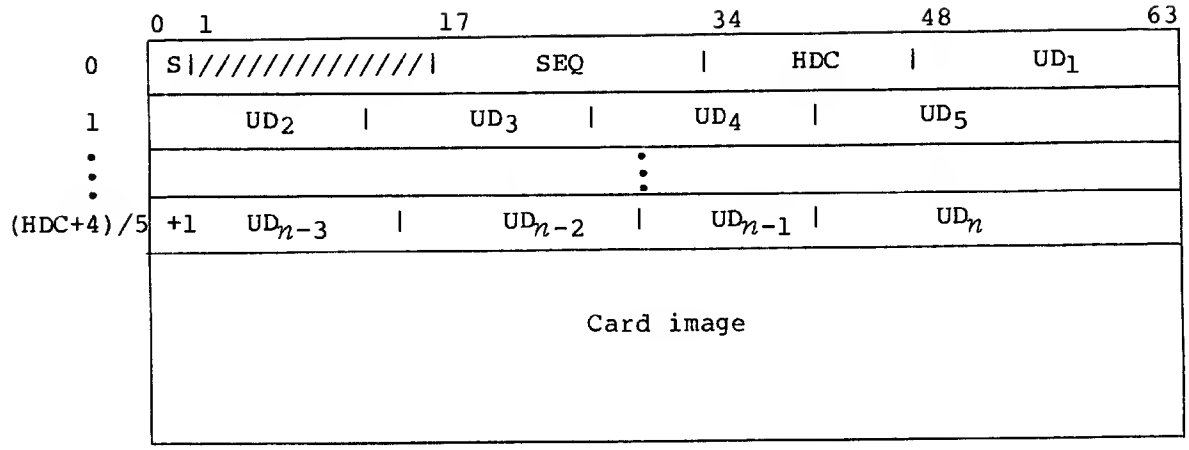


Figure C-2. PL format 2

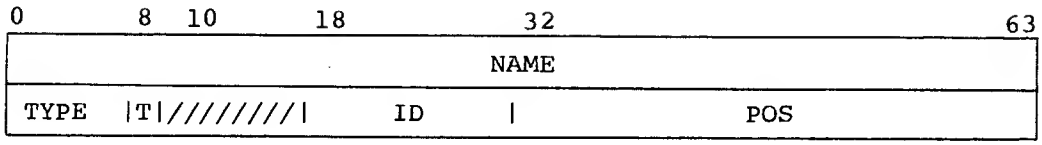
PL Card Format:



Field	Words	Bits	Description
S	0	0	Card status: 0 Active 1 Inactive
	0	1-16	Reserved
SEQ	0	17-33	Card sequence number
HDC	0	34-47	Header descriptor count
UD_1	0	48-63	First UPDATE descriptor (specifies name that introduces the card)
UD_i-UD_n	1-(HDC+4)/5 +1		Modification descriptors

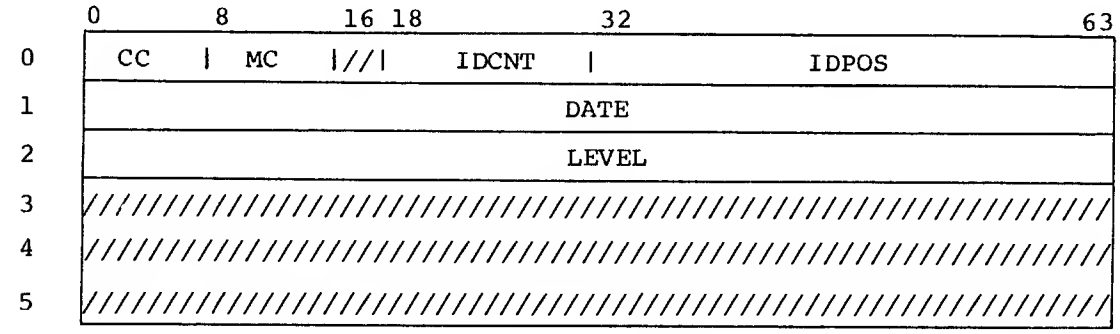
The format of each descriptor is identical to that of the corresponding format 1 PL fields.

Identifier Table Format:



Field	Bits	Description
NAME	0-63	Identifier name (left-adjusted, zero fill)
TYPE	0-7	Identifier type: 0 Modification 1 Deck 2 Common deck 3 Common deck/no propagation
T	8	Temporary flag (used internally, always 0 in PL)
	9-17	Reserved
ID	18-31	Identifier number
POS	32-63	Position of deck within PL (0 if TYPE=0)

PL Information File Format:



Field	Words	Bits	Description
CC	0	0-7	PL check character (O'142 = lowercase B)
MC	0	8-15	PL master character
	0	16-17	Reserved
IDCNT	0	18-31	Number of identifier table entries
IDPOS	0	32-63	PL position of start of identifier table
DATE	1	0-63	ASCII date of PL creation
LEVEL	2	0-63	Name of last identifier added to PL
	3-5	0-63	Reserved

UPDATE EXAMPLES D

The following sample jobs show program library creation, program library modification, adding new data to the library, extracting decks, removing decks, and library editing.

PL CREATION

This example shows the creation of program library PRLIB1. The PL consists of two decks - deck ABC and common deck XYZ. The P=0 entry in the UPDATE statement indicates that there is no existing PL. The omission of the other possible parameters indicates the standard defaults. (See section 2, The UPDATE Statement.)

```
JOB,JN=UP1.
UPDATE,P=0.
SAVE,DN=$NPL,PDN=PRLIB1.
  .
  .
  .
eof
*DK ABC
  (deck ABC here)
*CDK XYZ
  (deck XYZ here)
eof
```

The following examples show the creation of program library PRLIB2. The PL consists of decks ABC and XYZ. Each example contains a COMPILE directive.

In example 2, because of the S parameter, a source dataset that consists of deck ABC will be generated and saved. In example 3 a compile dataset that consists of deck ABC will be generated and compiled.

Example 2:

```
JOB,JN=UP2.
UPDATE,P=0,C=0,S.
SAVE,DN=$NPL,PDN=PRLIB2.
SAVE,DN=$SR,PDN=DSX.
  .
  .
  .
eof
*DK ABC
  (deck ABC here)
*CDK XYZ
  (deck XYZ here)
*C ABC
eof
```

Example 3:

```
JOB,JN=UP2.
UPDATE,P=0.
SAVE,DN=$NPL,PDN=PRLIB2.
CFT,I=$CPL.
   .
   .
   .
eof
*DK ABC
  (deck ABC here)
*CDK XYZ
  (deck XYZ here)
*C ABC
eof
```

In example 4, program library PRLIB3 is created. The PL consists of source dataset DS1 and deck XYZ. Note that dataset DS1 is a previously generated dataset that can contain any number of decks.

Example 4:

```
JOB,JN=UP3.
ACCESS,DN=DS1.
UPDATE,P=0.
SAVE,DN=$NPL,PDN=PRLIB3.
  .
  .
  .
eof
*READ DS1
*DK XYZ
  (deck XYZ here)
eof
```

Example 5 creates a new program library named $NPL that consists of two common decks and two regular decks. The library is saved as a permanent dataset named MYUPDATEPL.

Example 5:

```
JOB,JN=UPJOB1.
UPDATE,P=0,N,F.                    Full UPDATE:  Generate new program
                                   library.
SAVE,DN=$NPL,PDN=MYUPDATEPL.
eof                                End of control statement file
*CDK CA
  (Common deck CA)
*CDK CB
  (Common deck CB)
*DK A
  (Deck A)
*DK B
  (Deck B)
eof                                End of directives file
```

PL MODIFICATION

The following job (1) updates the previously generated library, (2) creates a compile dataset containing decks A through C and any common decks they call, and (3) generates an updated version of the program library. This library is saved as the next edition of permanent dataset MYUPDATEPL.

```
JOB,JN=UPJOB2.
ACCESS,DN=$PL,PDN=MYUPDATEPL.      Access highest edition number.
UPDATE,N.                          Compile dataset contents determined by
                                   C directives; new program library
                                   generated.
SAVE,DN=$NPL,PDN=MYUPDATEPL.       Save as next higher edition number.
eof                                End of control statement file
*ID MOD1                           Modification set named MOD1
*D A.15,A.20                       Replace cards 15 through 20 of deck A
                                   with new text cards.
  (Text cards)

*I B.119                           Insert cards after card 119 of deck B.
  (Text cards)
*DK C                              Introduce new deck.
  (Deck C)
*C A.C                             Write decks A and C and any common
                                   decks they call to compile dataset
                                   $CPL.  (Deck B is also written.)
eof                                End of directives file
```

The following job tests modification set MOD2. The changes are not permanently incorporated into the library (i.e., no new program library is generated and saved). The UPDATE list options are turned off. The contents of the compile dataset are determined by *C directives.

```
JOB,JN=UPJOB3.
ACCESS,DN=$PL,PDN=MYUPDATEPL.      Access highest edition number.
UPDATE,L=0,C=COMPILE.
CAL,I=COMPILE.
LDR.
REWIND,DN=$IN.
COPYF,I=$IN,O=MOD2.
REWIND,DN=MOD2.
SAVE,DN=MOD2.                      Save modification set MOD2.
eof                                End of control statement file
*ID MOD2
*D MOD1.2                          Replace card 2 of MOD1 modifications.
  (Text cards)
*B C.3                             Insert text cards before card 3 of
                                   deck C.
  (Text cards)
*C A.C                             Write decks, A, B, and C and any
                                   common decks they call onto compile
                                   dataset $CPL.
eof                                End of directives file
```

READ FROM ALTERNATE DATASET

MOD2 changes are introduced from datasets MOD2, DECK, and $IN. The contents of the compile dataset are determined by the Q option on the UPDATE statement.

```
JOB,JN=UPJOB4.
ACCESS,DN=MOD2,PDN=MOD2.
ACCESS,DN=DECK,PDN=DECK.
ACCESS,DN=$PL,PDN=MYUPDATEPL.      Access latest edition number.
UPDATE,Q=A:B:C:D,N.
CAL,I=$CPL.
LDR.
SAVE,DN=$NPL,PDN=MYUPDATEPL.       Save next higher edition.
eof
*READ MOD2                         Dataset MOD2 contains modification set
                                   MOD2 from previous example.
```

(Directives continued on next page.)

```
    *READ DECK                          Contents of dataset DECK:
                                            *DK D
                                               (Deck D)
                                            *CDK CC
                                               (Common deck CC)
                                          eof
*D C.12,C.16                            Delete cards 12 through 16 of deck C.
    *B B.17                             Insert the following text cards before
                                        card 17 in deck B.
      (Text cards)
    *CDK CD
      (Common deck CD)
    eof
```

INPUT DATASET NOT $IN

The following job adds to the library a common deck named CE and replaces lines of code in an existing deck with a call to CE. The input stream is on dataset UPIN. No compile dataset is generated.

```
JOB,JN=UPJOB5.
ACCESS,DN=$PL,PDN=MYUPDATEPL.
ACCESS,DN=UPIN,PDN=MYUPIN.
UPDATE,N,C=0,I=UPIN.
SAVE,DN=$NPL,PDN=MYUPDATEPL.
eof
```

The following are contents of directives dataset UPIN:

```
*ID MOD3
*CDK CE                                 Add common deck CE.
  (Common deck CE)
*D C.25,C.463
*CALL CE                                CALL directive inserted as text in
                                        place of deleted cards.
eof                                     End of directives file
```

EXTRACT DECKS FOR COMPILATION

The following job does not change the program library but extracts selected decks and any decks they call for compilation and inclusion on a source dataset ($SR). The master character is $, which was used when library FPL was created.

```
JOB,JN=UPLIB.
ACCESS,DN=FPL.
UPDATE,P=FPL,*=$,S,Q=SQRT:TANH:SIN:I=0.
CAL,I=$CPL.
SAVE,DN=$SR,PDN=MYFTN.
eof                                   End of control statement file
```

EXTRACT DECKS FOR COMPILATION (NO SOURCE)

A very common situation is the generation of a compile dataset which is a subset of the PL decks. No input is provided and no source input is desired. The following job illustrates the easiest way to perform this task.

```
JOB,JN=GET.
ACCESS,DN=$PL,PDN=COSPL.
UPDATE,I=0,Q=ST:CT:E:J:S.
          .
          .
          .
```

The decks selected (ST, CT, E, J, and S) are all written to $CPL, ready for assembly.

DECK REMOVAL AND POSITIONING

The following job illustrates the PURGEDK and MOVEDK directives in a typical application. The program library MYUPDATEPL currently consists of the following decks:

```
CDK  CA
CDK  CB
DK   A
DK   B
DK   C
CDK  CC
CDK  CD
DK   D
CDK  CE
```

This job replaces deck B with a new version in the same position relative to decks A and C:

```
JOB,JN=NEWB.
ACCESS,DN=$PL,PDN=MYUPDATEPL.
UPDATE,L=0,C=0,N,F.
SAVE,DN=$NPL,PDN=MYUPDATEPL.
eof
*PURGEDK B
*DK B
  (Text cards)
*MOVEDK B:A
eof
```

PL EDITING

As modifications to a PL accumulate, the user can reduce the size of the PL by permanently removing deactivated cards from one or more decks. This process is known as editing. The EDIT directive specifies a deck or group of decks to be edited.

```
JOB,JN=EDITPL.
ACCESS,DN=$PL,PDN=MYUPDATEPL.
UPDATE,C=0,N.
SAVE,DN=$NPL,PDN=MYUPDATEPL.
eof
*EDIT A.D                    Edit decks A through D.
*EDIT CA.CD                  Edit common decks CA through CD.
eof                          End of directives file
```

INDEX

INDEX

READERS COMMENT FORM

UPDATE Reference Manual SR-0013 D

Your comments help us to improve the quality and usefulness of our publications. Please use the space provided below to share with us your comments. When possible, please give specific page and paragraph references.

NAME _____

JOB TITLE _____

FIRM _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

CRAY RESEARCH, INC.

CUT ALONG THIS LINE

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO 6184 ST PAUL MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

Attention:
PUBLICATIONS

1440 Northland Drive
Mendota Heights, MN 55120
U.S.A.

FOLD

STAPLE